Safe trajectory optimization for whole-body motion of humanoids

Valerio Modugno, Gabriele Nava, Daniele Pucci, Francesco Nori, Giuseppe Oriolo, Serena Ivaldi

Safe trajectory optimization for whole-body motion of humanoids

Valerio Modugno[1,2], Gabriele Nava[3], Daniele Pucci[3], Francesco Nori[3], Giuseppe Oriolo[1], Serena Ivaldi[2]

***Abstract*— Multi-task prioritized controllers generate complex behaviors for humanoids that concurrently satisfy several tasks and constraints. In our previous work we automatically learned the task priorities that maximized the robot performance in whole-body reaching tasks, ensuring that the optimized priorities were leading to safe behaviors. Here, we take the opposite approach: we optimize the task trajectories for whole-body balancing tasks with switching contacts, ensuring that the optimized movements are safe and never violate any of the robot and problem constraints. We use (1+1)-CMA-ES with Constrained Covariance Adaptation as a constrained black box stochastic optimization algorithm, with an instance of (1+1)-CMA-ES for bootstrapping the search. We apply our learning framework to the prioritized whole-body torque controller of iCub, to optimize the robot's movement for standing up from a chair.**

I. INTRODUCTION

The generation of complex whole-body movements for humanoid robots typically involves the definition of a set of tasks (i.e., tracking a desired trajectory for the joints or the end-effectors, either positions or forces), performance objectives (e.g., minimize the torques), under a set of constraints (e.g., joint limits, motors limits) that assure that the motion is physically feasible on the real robot. For example, let us consider the humanoid iCub (Fig.1) that must stand up from a chair. This motion, trivial for a human, is very challenging to be realized for a humanoid. During the execution of the stand up we want to be sure that the robot produces the right acceleration of its Center of Mass (CoM) while balancing. This necessitates optimizing its posture at each time step. At the same time we want to guarantee the feasibility and the safety[1] of the generated motions, i.e., we must take into account torque and joint limits, collisions, external forces, balancing constraints to prevent falling *etc.*

To solve this multi-task, multi-constraint problem, the research community has converged to a consensus framework in the last years: that is by using prioritized multi-task controllers, either with strict task priorities [1] or soft task priorities (also called weights) [2]. Typically, they are formulated as Quadratic Programming (QP) problems [3].

*This work was supported by the European Projects CoDyCo (FP7, n.600716), Comanoid (H2020, n.645097) and An.Dy (H2020, n.731540)

[1] Sapienza Università di Roma, Italy.
{modugno@dis,oriolo@diag}.uniroma1.it
[2] Inria Nancy - Grand Est, CNRS & Univ. Lorraine, Loria, France.
serena.ivaldi@inria.fr
[3] Fondazione Istituto Italiano di Tecnologia, Italy.
firstname.lastname@iit.it

[1]By safety, here we mean the satisfaction of all the mechanical constraints of the robot, task constraints that guarantee that motions are feasible and do not make the robot fall, and constraints deriving from the interaction with the environment.

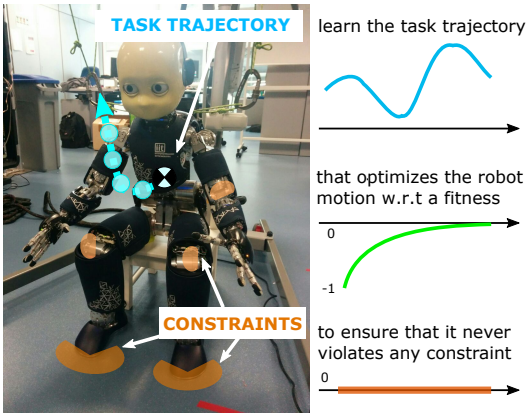

Fig. 1: The humanoid robot iCub performing a whole-body motion (*stand-up from the chair*), with several tasks and constraints. In this paper we optimize the desired task trajectory w.r.t. a fitness function, guaranteeing that the global robot behavior is safe: it never violates any of the constraints.

The main problem with these methods is that when the global movement to realize is complex, like in the stand-up case, a significant amount of manual tuning is required to optimize the motion and adapt it to the robot, as one cannot guarantee that the QP controller can successfully execute any desired trajectory while satisfying all the constraints, not to mention the possibility that the QP solver uses constraints relaxation to find a solution. Generally, one can optimize the task priorities/weights, their evolution in time (e.g., their relative priority, or scalar value, or the task transitions), and/or the task trajectories, by acting on the desired trajectories (e.g., desired CoM) or some via-points (e.g., to avoid collisions or potentially unstable configurations).

In previous work [4] we proposed to automatically learn the task priorities by stochastic optimisation. Our hypothesis was that the tasks were fixed, but not the task weights. In [5] we proposed to use constrained optimization to find solutions that never violate constraints, guaranteeing a safe trajectory and control generation.

In this paper, we take the opposite approach: we consider a controller with a fixed strategy and task priorities, and optimize the task trajectories, i.e. the reference trajectories or desired trajectories of the tasks. We do not use a weighted QP controller but the prioritized whole-body controller proposed by Nava *et al.* [6], which is capable of performing highly dynamic tasks [7]. We parametrize the task trajectories, and use constrained black-box stochastic optimization to ensure that the optimized trajectories are "safe" and that they never violate any constraint. As in [5], we rely on (1+1)-CMA-ES

with CCA to find solutions that do not violate constraints, but we improve the search bootstrap phase. To show the effectiveness of our approach, we show how we can optimize a particular motion for iCub: *standing-up from a chair*.

This kind of motion is characterized by switching contacts (from the legs on the chair to the feet on the ground), physical interaction with the environment and several balance constraints to be verified. Executing this kind of motion with the previous approach (learning task priorities) is hardly possible, because it is not suited to segment a complex movement into phases characterized by different contacts configurations.

The paper is organized as follows. Section II presents the related research in safe trajectory optimization for humanoid robots. Section III formulates our learning problem and describes the framework for learning task trajectories for controlling redundant robots like humanoids, including the whole-body controller and the constrained optimization algorithms retained for the study. Section IV illustrates the experiments with the iCub humanoid robot.

II. RELATED WORK

The problem of whole-body control for redundant robots such as humanoids involves fulfilling multiple operational, posture and force tasks, while satisfying several constraints that guarantee the physical feasibility of the generated motions [8], [9]. The problem is classically solved by prioritized multi-task controllers with strict task priorities [1], [10] or soft task priorities (also called weights) [2], [11]. It is formulated as a QP problem subject to constraints, that is solved at each time step of the control loop [3], [12].

To optimize the global robot movement, one classic approach is to fix the desired task trajectories (e.g., they are known in advance, or demonstrated by a human) and optimize the task priorities/weights (including their value in time, their transitions, *etc.*). Very frequently, this optimization is done manually. In [4], [5] we parametrized the task priorities and optimized their parameters to obtain behaviors that were maximizing a fitness function while satisfying the problem constraints.

Another approach consists in fixing the task priorities and optimize the desired task trajectories (in the form of set-points, via-points or entire trajectories). For example, in [13] the authors parametrized the task trajectories of a QP controller with a set of waypoints, and used Bayesian Optimization to optimize the global robot's movement reducing a task cost.

Trajectory optimization has been a central area of research in robotics for decades, especially to generate highly dynamic motions. In some cases, the problem of trajectory optimisation has been decoupled from the controller used to execute the trajectory on the robot. For example, in [14] Mordatch *et al.* propose the Contact Invariant Optimization method to synthesize highly complex behaviours, decomposed in different phases with their own set of contacts and weight functions. In [15] Park *et al.* propose the Incremental Optimization for Real-Time Replanning (ITOMP), an optimal trajectory planning method that can avoid moving obstacle by interleaving planning and execution.

Both in [14] and [15], the problem's constraints are managed as an additional cost inside the objective function, which might lead to solutions that are not feasible.

An alternative approach is to reframe the trajectory optimization problem as an optimal control one. For locally linearized systems with quadratic cost function and Gaussian noise, one of the most prominent methods is the iLQG [16]: it iteratively improves an optimal controller that locally optimizes a linearisation of a general non-linear system around the current trajectory. In [17] ILQG was used as optimization routine inside a Model Predictive Control scheme. In [18] Tassa *et al.* propose a generalized version of Differential Dynamic Programming to provide a second order approximation of the Bellman equation, (while ILQG employs a first order approximation of the same expression). These trajectory optimization methods provide explicit means to manage constraints on the control inputs with backtracking line search in [16], [17], or by solving quadratic program with box constraints for each time step in [18]. None of them combat the state space constraint issue in an explicit way, so there is no guarantee of constraint satisfaction in scenarios with real robots. Toussaint proposed a probabilistic extension of ILQG that it can deal with non-Gaussian noise models but relaxes more the constraints [19]. More recently, Akrour *et al.* in [20] extended [16] by dropping the linear approximation of the model dynamics, and replacing it with a model free approach based on a quadratic Q-function. However, as the previous methods, it ignores the existence of constraints defined in the state and action spaces.

Reist *et al.* propose a framework that combines optimal control and trajectory planning to compute a set of policies that stabilize from any starting point a non-linear system towards a goal state, while taking into account the control-state constraints [21]. The motion planning module generates an open loop trajectory for the system that satisfies the control-state constraints. Then the feedback module ensures local stability around the nodes of the computed trajectory. The union of the controllable region around each ball produce the so called "funnel". All the states in the funnel under the optimal control policy reach the goal and satisfy all the problem's constraints. It has two main limitations: first, it only allows quadratic cost functions; second, it cannot be easily extended to control problems with switching contacts.

III. METHOD

The framework proposed in this paper addresses all the issues of the other methods presented in Section II. Our learning module relies on a derivative-free constrained optimization method that poses no restrictions on the structure of the problem and guarantees constraint satisfaction both in the state and in the control space. The integration of the learning module with an established feedback torque controller [6], [7] let us obtain solutions that are robust and that can be easily extended to multi-contact scenarios.

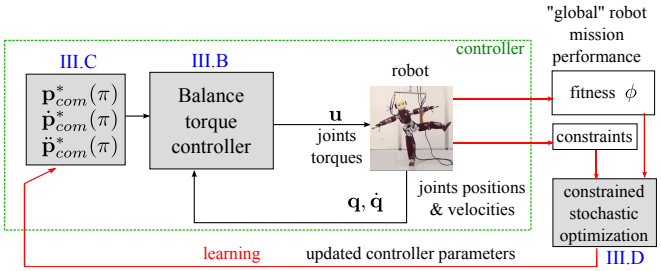

Fig. 2: Overview of the proposed method. The balance controller executes a desired task trajectory, represented by a parametrized function. An outer learning loop enables the optimization of the parameters of the task trajectory, taking into account the constraint violations in an explicit way.

Figure 2 outlines our method. The balance torque controller of [6], [7] tracks the desired task trajectory (i.e., the CoM) sending joint torque commands to the robot (III-B). The task trajectories $\mathbf{p}(\pi)$ are parametrized functions, where π are the free parameters to be optimized (III-C). At the end of a roll-out (i.e., execution of the *stand-up from the chair*) we evaluate the fitness of the movement and the constraint violation. A constrained stochastic optimisation algorithm updates the parameters of the task trajectories (III-D).

A. Notation

Throughout the paper we use the following definitions:

- ${}^a(\cdot)_b$: point b expressed in frame a
- I: inertial frame
- ${}^I\mathbf{p}_B, {}^IR_B$,: base position and orientation in I frame
- $\mathbf{q}_j$: joint positions
- $\mathbf{M}, \mathbf{C}, \mathbf{G}$: dynamic matrices and vectors of the floating base robot
- $\mathbf{J}_{C_k}$: $k-th$ contact Jacobian
- $\mathbf{J}_{(\cdot)}$: robot Jacobian
- $\mathbf{f}$: external wrench
- $\mathbf{h}$: centroidal momentum
- $\phi(\cdot)$: objective function
- n_{IC}: # of inequality constraints $g_i(\cdot)$
- n_{EC}: # of equality constraints $h_i(\cdot)$
- $n_C = n_{IC} + n_{EC}$: total number of constraints
- $\Pi \subseteq \mathbb{R}^{n_P}$: parameter space of the RBFs network
- $\pi_k \in \Pi$: k-th candidate at the current generation
- K: total number of candidates for each generation
- K_e: number of best candidates or *elites*
- $\pi_{1:K_e}$: best candidates of the current generation
- $\mathcal{N}(\bar{\pi}, \Sigma)$: Gaussian distribution with mean $\bar{\pi}$ and covariance Σ
- σ^2: step size
- $\hat{l}(\pi_k)$: penalty factor
- $\mathbb{1}_{\{\cdot\}}$ indicator function

B. Whole-body torque controller

We hereby describe the balance torque controller. Let us consider a dynamic representation of a floating base robot. We represent the configuration space of a humanoid robot by a triplet $\mathbf{q} = ({}^I\mathbf{p}_B, {}^IR_B, \mathbf{q}_j) \in \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n$ where ${}^I\mathbf{p}_B$ and ${}^IR_B$ describe respectively the position and orientation of the base of the humanoid (usually located in the torso) respect of an inertial reference frame, while $\mathbf{q}_j$ represents the vector of joint angles. The total Degrees of Freedom (DoF) of the robot are $n+6$ because the free floating base adds 6 DoF. The generalized velocity associated to the system is $\nu = ({}^I\dot{\mathbf{p}}_B, {}^I\omega_B, \dot{\mathbf{q}}_\mathbf{j}) = (\mathbf{v}_B, \dot{\mathbf{q}}_j)$ where $\mathbf{v}_B$ describes the spatial velocities of the base frame with respect to the inertial frame; ${}^I\omega_B$ is defined as $[{}^I\omega_B]_\times = {}^I\dot{R}_B {}^IR_B^T$, with $[\cdot]_\times$ being the screw symmetric matrix obtained from ${}^I\omega_B$. With the hypothesis of a flat contact surface (reasonable hypothesis in our case, since our contacts are on a flat ground and flat chair) the constrained dynamic model describing the robot is:

$$0 = \mathbf{J}_{C_k}\nu \tag{1}$$

$$\mathbf{Su} = \mathbf{M}(\mathbf{q})\dot{\nu} + \mathbf{C}(\mathbf{q},\nu)\nu + \mathbf{G}(\mathbf{q}) - \sum_k \mathbf{J}_{C_k}\mathbf{f}_k \tag{2}$$

where $\mathbf{u} \in \mathbb{R}^n$ denotes the generalized force at the joints, $\mathbf{S}$ is a selection matrix, $\mathbf{M} \in \mathbb{R}^{n+6} \times \mathbb{R}^{n+6}$ is the generalized mass matrix, $\mathbf{C}(\mathbf{q},\nu)$ and $\mathbf{G}(\mathbf{q})$ account respectively for the non inertial forces and the gravity torque. $\mathbf{f}_k \in \mathbb{R}^6$ represents the external wrench and is related to the Ground Reaction Forces (GRF) and the Center of Pressure (CoP) while the Jacobian $\mathbf{J}_{C_k}$ maps the robot velocity to the linear and angular velocity of the frame where the external wrench is exerted:

$$\mathbf{J}_{C_k}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}^b_{C_k}(\mathbf{q}) & J^j_{C_k}(\mathbf{q}) \end{bmatrix} \tag{3}$$

$$\mathbf{J}^b_{C_k} = \begin{bmatrix} \mathbf{1}_3 & -[{}^I\mathbf{p}_{C_k} - {}^I\mathbf{p}_B]_\times \\ \mathbf{0}_3 & \mathbf{1}_3 \end{bmatrix} \tag{4}$$

The dimension of $\mathbf{J}_{C_k}$ depends on the number and the kind of contacts with the environment. For example a double stance contact that constraints both orientation and position of the contact point has 12 dimensions (6 for each foot in contact with the ground). Equation 1 is equal to zero because we make the hypothesis of no slippage on the contact surfaces. Due to the fact that a free floating model is underactuated, we can rewrite the equation 2 into two different parts:

$$\mathbf{0} = \mathbf{J}_{C_k}\dot{\nu} + \dot{\mathbf{J}}_{C_k}\nu \tag{5}$$

$$\mathbf{0} = \begin{bmatrix} \mathbf{M}_b(\mathbf{q}) & \mathbf{M}_{bj}(\mathbf{q}) \end{bmatrix}\dot{\nu} + \mathbf{C}_b(\mathbf{q},\nu)\nu + \mathbf{G}_b(\mathbf{q}) - \textstyle\sum_k \mathbf{J}^b_{C_k}\mathbf{f}_k \in \mathbb{R}^6 \tag{6}$$

$$\mathbf{u} = \begin{bmatrix} \mathbf{M}_{bj}(\mathbf{q})^T & \mathbf{M}_j(\mathbf{q}) \end{bmatrix}\dot{\nu} + \mathbf{C}_j(\mathbf{q},\nu)\nu + \mathbf{G}_j(\mathbf{q}) - \textstyle\sum_k \mathbf{J}^j_{C_k}\mathbf{f}_k \in \mathbb{R}^n \tag{7}$$

with $\mathbf{M}_b(\mathbf{q}) \in \mathbb{R}^{6\times 6}$ $M_j(\mathbf{q}) \in \mathbb{R}^{n\times n}$ and $\mathbf{M}_{bj}(\mathbf{q}) \in \mathbb{R}^{6\times n}$. It is proven that it is possible to diagonalize the generalized mass matrix $\mathbf{M}$ leading to a decoupling of base and joint accelerations [22]. We introduce the following change of coordinates $\mathbf{q} = \mathbf{q}$, $\overline{\nu} = \mathbf{T}(\mathbf{q})\nu$

$$\mathbf{T} = \begin{bmatrix} {}^c\mathbf{X}_B & {}^c\mathbf{X}_B\mathbf{M}_b^{-1}\mathbf{M}_{bj} \\ \mathbf{0}_{n\times 6} & \mathbf{1}_n \end{bmatrix} \tag{8}$$

$${}^c\mathbf{X}_B = \begin{bmatrix} \mathbf{1}_3 & -[{}^I\mathbf{p}_c - {}^I\mathbf{p}_B]_\times \\ \mathbf{0}_{3\times 3} & \mathbf{1}_3 \end{bmatrix} \tag{9}$$

where the c superscript represents the frame centered in the center of mass of the floating base platform, oriented as the inertial frame I. Equation 2 becomes:

$$\mathbf{Su} = \overline{\mathbf{M}}(\mathbf{q})\dot{\overline{\nu}} + \overline{\mathbf{C}}(\mathbf{q},\overline{\nu})\overline{\nu} + \overline{\mathbf{G}} - \sum_i \mathbf{J}^T_{C_i}\mathbf{f}_i \tag{10}$$

where

$$\overline{\mathbf{M}} = \begin{bmatrix} \overline{\mathbf{M}}_b(\mathbf{q}) & \mathbf{0}_{6\times n} \\ \mathbf{0}_{n\times 6} & \overline{\mathbf{M}}_j(\mathbf{q}_j) \end{bmatrix} \tag{11}$$

$$\overline{\mathbf{M}}_b(\mathbf{q}) = \begin{bmatrix} m\mathbf{1}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \tilde{\mathbf{I}}(\mathbf{q}) \end{bmatrix} \tag{12}$$

$$\overline{\mathbf{G}} = mg\,\mathbf{e}_3 \tag{13}$$

$$\overline{\mathbf{J}}_{C_i}(\mathbf{q}) = \begin{bmatrix} \overline{\mathbf{J}}^b_{C_i}(\mathbf{q}) & \overline{\mathbf{J}}^j_{C_i}(\mathbf{q}) \end{bmatrix} \tag{14}$$

$$\overline{\mathbf{J}}^b_{C_i} = \begin{bmatrix} \mathbf{1}_3 & -[{}^I\mathbf{p}_{C_i} - {}^I\mathbf{p}_c]_\times \\ \mathbf{0}_{3\times 3} & \mathbf{1}_3 \end{bmatrix} \tag{15}$$

with m the mass of the robot and $\tilde{\mathbf{I}}$ the inertia of the robot defined in the CoM frame. Due to the change of coordinate, the floating base velocities now represent the linear and angular velocities of the CoM. In particular the angular velocity is the *average angular velocity*. Equation 10 unifies in one expression the centroidal dynamics and the dynamic description of the free floating base. From now on all the subsequent equations will refer to Equation 10 and to lighten the notation we will drop the overline sign. The change of coordinate allows for a simple computation of the robot momentum. From [23] and Equation 10,the robot momentum is defined as $\mathbf{h} = \mathbf{M}_b\mathbf{v}_b$.The D'Alembert's principle states that the rate of change of the robot momentum depends on the external forces acting on the robot (force from the environment and gravity in the CoM). From Equation 10 and d'Alembert's principle we can write:

$$\dot{\mathbf{h}} = \frac{d}{dt}(\mathbf{M}_b\mathbf{v}_B) = \mathbf{J}_B^T\mathbf{f} - mg\,\mathbf{e}_3 \tag{16}$$

By inverting the previous relation we can find an expression that connects the desired rate of the change of the centroidal momentum and the external forces:

$$\mathbf{f} = \mathbf{J}_B^{-T}(\dot{\mathbf{h}}^* + mg\,\mathbf{e}_3) \tag{17}$$

$$\dot{\mathbf{h}}^* = \dot{\mathbf{h}}^{des} + \mathbf{K}_p(\mathbf{h} - \mathbf{h}^d) + \mathbf{K}_i\int \mathbf{h} - \mathbf{h}^d \tag{18}$$

where $\dot{\mathbf{h}}^*$ combines a feed-forward term and a Proportional Integral correction to ensure that the system achieves the desired centroidal momentum. The control torques that realize the instantaneous external forces and satisfy the holonomic constraints at the contact point are:

$$\mathbf{u} = \Lambda^\dagger(\mathbf{J}_C\mathbf{M}(\mathbf{F} - \mathbf{J}_C^T\mathbf{f}) - \dot{\mathbf{J}}_C\nu)) + \mathbf{N}_\Lambda\mathbf{u}_0 \tag{19}$$

with $\Lambda = \mathbf{J}_j\mathbf{M}_j$, $(\cdot)^\dagger$ the pseudo-inversion, $\mathbf{F} = \mathbf{C} + \mathbf{G}$ and $\mathbf{u}_0 = \mathbf{F} - \mathbf{J}_j\mathbf{f} + \mathbf{K}_p^j(\mathbf{q}_j - \mathbf{q}_j^{des}) + \mathbf{K}_d\dot{\mathbf{q}}_\mathbf{j}$ the posture task projected in the null space of the primary task that is introduced for the stability of the zero dynamics. In [6] the authors empirically proved that in a centroidal momentum controller the posture task do not always assure the internal stability. To overcome this issue in Equation 18 they replace the momentum error with

$$\mathbf{h}_\mathbf{err} = \begin{bmatrix} \overline{\mathbf{J}}^l_G(\mathbf{q}_j) \\ \overline{\mathbf{J}}^k_G(\mathbf{q}^d_j) \end{bmatrix}\dot{\mathbf{q}}_j \tag{20}$$

where $[\overline{\mathbf{J}}^l_G(\mathbf{q}_j)\overline{\mathbf{J}}^k_G(\mathbf{q}^d_j)]$ comes from a reduced expression of the centroidal momentum matrix that maps the joints velocities to the velocities of the floating base and is defined as $\overline{\mathbf{J}}_G(\mathbf{q}_j) = -\mathbf{M}_b\mathbf{J}_b^{-1}\mathbf{J}_j$. This substitution is motivated by the Lyapunov stability of the linearized closed loop dynamics around the joint positions-velocities equilibrium point $(\mathbf{q}^d_j, 0)$. Equation 17 is mathematically sound only if the robot is standing on one foot and we ignore all the balance related constraints (GRF contained inside the friction cone and CoP inside the support polygon). To resolve the redundancy in the external forces in a multi-contact scenario and to take into account all the conditions for a stable stance we can recast the computation of the external forces as an optimization problem. This is done in our case, since the robot transitions between 2 contact forces in the legs when seated (contacts between legs and chair) and 2 in the feet when it stands up (contacts between feet sole and ground).

C. Trajectories Parametrization

In our application, our desired task trajectory is the one of the CoM. The CoM Cartesian trajectory is modelled as a weighted sum of normalized Radial Basis Functions (RBFs):

$$p^{*,i}_{com}(\pi_i, t) = \frac{\sum_{k=1}^{n_r}\pi_{ik}\psi_k(\mu_k, \sigma_k, t)}{\sum_{k=1}^{n_r}\psi_k(\mu_k, \sigma_k, t)} \tag{21}$$

where $p^{*,i}_{com}$ is the desired profile in time of one of the component of the final CoM trajectory, $\psi_k(\mu_k, \sigma_k, t) = \exp\left(-1/2[(t-\mu_k)/\sigma_k]^2\right)$, with fixed mean μ_k and variance σ_k of the basis functions, n_r is the number of RBFs and $\pi_i = (\pi_{i1}, \ldots, \pi_{in_r}) \subseteq \mathbb{R}^{n_p}$ is the set of parameters for each trajectory dimension. Generally for the design of Cartesian trajectory some conditions must be verified, for example passing through a set of waypoints or imposing particular conditions on initial or final velocities and accelerations. In this work, due to the application scenario, we employed an extended version of the RBF that allows for waypoints. We augment the RBF model by adding a set of RBFs ψ_j, one for each waypoint, located in correspondence of the time μ^c_j in which we impose the waypoint passage:

$$p^{*,i}_{com}(\pi_i, t) = \frac{\sum_{j=1}^{n_{wp}}\pi^c_{ij}\psi_j(\mu^c_j, \sigma_j, t) + \sum_{k=1}^{n_r}\pi^f_{ik}\psi_k(\mu_k, \sigma_k, t)}{\sum_{k=1}^{n_r+n_{wp}}\psi_k(\mu_k, \sigma_k, t)} \tag{22}$$

Once we set the value of the free parameters π^f_i, we need to compute the value of the fixed parameters in order to match the waypoint passage conditions. Given the vector of values $\mathbf{f^{wp}} = [\mathbf{f_1}, \ldots, \mathbf{f_{n_{wp}}}]$ that defines the value of the function $p^{*,i}_{com}$ at each way-point, we can write:

$${\pi_i}^c = \mathbf{A}^{-1}\mathbf{b} \tag{23}$$

where

$$\mathbf{A} = \begin{bmatrix} \psi_1(\mu^c_1, \sigma_1, \mu^c_1) & \cdots & \psi_{n_{wp}}(\mu^c_{n_{wp}}, \sigma_{n_{wp}}, \mu^c_1) \\ \vdots & & \vdots \\ \psi_1(\mu^c_1, \sigma_1, \mu^c_{n_{wp}}) & \cdots & \psi_{n_{wp}}(\mu^c_{n_{wp}}, \sigma_{n_{wp}}, \mu^c_{n_{wp}}) \end{bmatrix} \tag{24}$$

$$\mathbf{b} = \begin{bmatrix} f^{wp}_1\sum_{k=1}^{n_r}\hat{\pi}^f_{ik}\psi_k(\mu_k, \sigma_k, \mu^c_1) - \sum_{k=1}^{n_r+n_{wp}}\psi_k(\mu_k, \sigma_k, \mu^c_1) \\ \vdots \\ f^{wp}_{n_{wp}}\sum_{k=1}^{n_r}\hat{\pi}^f_{ik}\psi_k(\mu_k, \sigma_k, \mu^c_{n_{wp}}) - \sum_{k=1}^{n_r+n_{wp}}\psi_k(\mu_k, \sigma_k, \mu^c_{n_{wp}}) \end{bmatrix} \tag{25}$$

For the case of exponential radial basis functions the matrix A is always invertible due to its particular structure.

D. Constrained Stochastic Optimisation

In the following section we formalize and describe our method for learning the parameters of the task trajectory with constrained stochastic optimisation. Given a fitness function $\phi(\pi) : \mathbb{R}^{n_P} \to \mathbb{R}$ and a set of equality and inequality constraints g, h, we want to find an optimal solution $\pi \in \Pi \subseteq \mathbb{R}^{n_P}$ to the problem:

$$\pi^{\circ} = \arg\max_{\pi} \phi(\pi) \tag{26}$$

$s.t.$

$$g_i(\pi) \leq 0 \quad i = 1, \ldots, n_{IC} \tag{27}$$

$$h_i(\pi) = 0 \quad i = 1, \ldots, n_{EC} \tag{28}$$

In our previous work [4], [5], we solve the problem with derivative-free methods, imposing no constraints on the structure of the fitness function and of the constraints. We use the known extensions of CMA-ES [24] that we benchmarked in [5] for learning task priorities, to harness the known advantage of having to tune few parameters.

For CMA-ES a single iteration (called *generation*) comprises several stages. From a multivariate Gaussian distribution $\mathcal{N}(\bar{\pi}, \sigma^2\Sigma)$ K samples π_k are drawn with a σ^2 step size; an evaluation of the objective function ϕ, called *fitness*, is performed for each sample π_k. Based on the fitness, the samples are ranked and the Gaussian distribution is updated according to the best K_e candidates $\pi_{1:K_e}$, called *elites*.

The update changes the mean, the covariance and the step size of the search distribution. The update for σ^2 and Σ exploits information both from last generation and all the previous ones. The step size update plays a key role to avoid premature convergence of the algorithm around local minima. A set of probability weights influences the mean update with one P_k associated to each elite candidate. a default value for each weight is $P_k = \ln(0.5(K_e+1)) - \ln(k)$. The initialization of CMA-ES requires the user to specify the value of the exploration rate, a number between 0 and 1, that affects the first value of the covariance matrix.

The classical CMA-ES is not designed to solve constrained black-box optimization problems, but there exist several variants of it in the literature that handle constraints. In [5] we benchmarked different ones and we showed that (1+1)-CMA-ES with Covariance Constrained Adaptation (CCA) [25] was the most appropriate for our robotics problems with several parameters and constraints. So we adopt it here. The constraints satisfaction is ensured by combining (i) a rejection rule that prevents the mean update when at least one constraint is violated and (ii) a different exploration strategy that exploits the information from violated constraints to change the covariance and steer the optimization inside the feasible region. In its basic version (1+1)-CMA-ES shows small differences with CMA-ES. In every generation only one sample (π_1, therefore $K = 1$) is drawn from the search distribution according to the following rule: $\pi_1 = \pi + \sigma\mathbf{D}\mathbf{z}$, where $\mathbf{D}$ is the Cholesky factor of the covariance matrix $\Sigma = \mathbf{D}^T\mathbf{D}$ and $\mathbf{z}$ is a standard normal distributed vector $\mathbf{z} \sim \mathcal{N}(0, I)$. The information about the successful steps are stored in a *search path* $\mathbf{s} \in \mathbb{R}^{n_P}$. For a detailed description of the update procedure of the original algorithm please refer to [26]. In [25] the authors kept in place the original update scheme when the candidate belongs to the free region while adding a new update procedure for the constraints violation case. To reduce the probability that the candidate may belong to the infeasible region, the authors propose to reduce the covariance matrix in the direction that is orthogonal to the constraint boundary. Each time the j-th constraint is violated, we update the corresponding constraint vector trace $\mathbf{v_j} \in \mathbb{R}^{n_P}$ and the matrix $\mathbf{D}$ according to:

$$\mathbf{v}_j^{new} = (1 - c_c)\mathbf{v}_j + c_c\mathbf{D}\mathbf{z}$$

$$\mathbf{D}^{new} = \mathbf{D} - \frac{\beta}{\sum_{j=1}^{n_C} \mathbb{1}_{\{g_j(\pi_1)>0)\}}} \sum_{j=1}^{n_C} \mathbb{1}_{\{g_j(\pi_1)>0)\}} \frac{\mathbf{v}_j\mathbf{w}^T}{\mathbf{w}^T\mathbf{w}}$$

where c_c and β are constants that control the update step respectively for $\mathbf{v}_j$ and $\mathbf{D}$, $\mathbf{w}_j = \mathbf{D}^{-1}\mathbf{v}_j$ and $\mathbb{1}_{\{g_j(\pi_1)>0)\}}$ is equal to one when $g_j(\pi_1) > 0$ and zero otherwise.

To summarize, the method tests one sample every generation and extends the covariance update to take into account the localization of the infeasible region in a neighborhood of the search distribution. The algorithm proceeds to update the mean of the search distribution only if the current candidate have a better fitness and the new sample belongs to the feasible region; these two conditions guarantees that the solution of the optimization problem always stays outside the infeasible region.

This method requires the starting point to satisfy all constraints in order to function properly. It means that we cannot initialize the method from scratch with random values but we need to provide a feasible starting point. To overcome this known limitation in this work we provide an extension of the current method with the introduction of a new step in the algorithm. In order to find a feasible starting point we start the algorithm with an unconstrained optimization problem with the subsequent fitness:

$$\phi_{bs}(T, n_c) = \begin{cases} -\sum_t^T \sum_{i=1}^{n_C} |\hat{e}(t,i,\pi)|, & \text{if constraints violation} \\ -\sum_t^T \sum_{i=1}^{n_C} |\hat{s}(t,i,\pi)|, & \text{if no constraints violation} \end{cases}$$

where T is the number of control steps, $\hat{e}(t,i,\pi) = \mathbb{1}_{\{g_i(\pi)>0\}}\, g_i(\pi)$ and $\hat{e}(t,i,\pi) = \mathbb{1}_{\{h_i(\pi)\neq 0\}}\, h_i(\pi)$ respectively for the inequality and equality constraints that are not satisfied at time t and $\hat{s}(t,i,\pi) = \mathbb{1}_{\{g_i(\pi)<=0\}}\, g_i(\pi)$, $\hat{s}(t,i,\pi) = \mathbb{1}_{\{h_i(\pi)=0\}}\, h_i(\pi)$ represent all the satisfied constraints at time t. To summarize during an experiment we sum all the constraints violations while ignoring all the satisfied one. Only if the candidate is feasible throughout the experiment, we sum the total amount of constraints satisfaction.

IV. EXPERIMENTS

In this section we present our experiments with the iCub humanoid performing a "*stand-up from the chair*" movement. All experiments are simulated in Gazebo. Our method is capable of optimizing the task trajectories to generate a safe

Posture	Torso (deg)	Larm (deg)	Lleg (deg)
$\mathbf{q}_{si}$	60,0.62,0.40	-67.2,34.1,4.8,43.2	84.3,0.8,0.1,-99.2,-15.8,0.1
$\mathbf{q}_{si}$	-10,0,0	-20,30,0,45	25.5,0,0,-18.5,-5.5,0

TABLE I: The intermediate ($\mathbf{q}_{si}$) and the final joint pose ($\mathbf{q}_{st}$) that define the secondary task.

motion even when the robot is switching contacts in physical interaction with the environment.

Robotics setup - Our experiments were performed with the iCub humanoid robot. iCub is a 53-DOF humanoid robot [27], but only 23-DOF are torque-controlled and used for balancing tasks. It is equipped with 6 force/torque sensors, placed in the middle of arms, legs and feet, that are used to compute the whole-body dynamics [28]. We developed our controller in Matlab/Simulink using the WBToolbox [29], which can be applied to both the simulated and real robot, whereas the learning and trajectory optimisation is done with our framework for learning task priorities [5]. Considering the high chance of breaking the real robot during learning, we perform all the learning procedure in simulation. The rollouts are performed in Gazebo.

Stand-up problem - We apply our method to solve the problem of finding a safe trajectory for the robot to stand up from a chair, where "safe" means physically feasible and that it does not violate any problem/robot constraint. The goal of the experiment is to learn an optimal CoM trajectory to move the robot from a sitting position to a double support stance that guarantees constraints satisfaction and minimizes the risk of falling. The desired trajectory is executed by the centroidal momentum controller described in Section III-B. For this experiment we constrain the desired CoM trajectory to lay on the sagittal plane of the robot (x,z plane in the robot world frame). For each trajectory component, we associate a RBFs network with $n^f = 5$ and we add two fixed viapoints for the starting and the final CoM positions. Then we use a third RBFs network to model the time law. To allow for the stand up movements in the controller, we need to switch from the lower back/legs contact points (when the robot is sitting on the chair) to the feet contact points. Therefore we introduce a switching time t_{switch}. This induces a natural segmentation of the movement in two distinct phases: the first is when the robot is sitting, the second is when the robot breaks the contacts with the chair and starts to stand up to reach the standing position. For the posture task we set two different keyframe joint postures: one at the end of sitting phase ($\mathbf{q}_{si}$) and one a the end of the sit up movement ($\mathbf{q}_{st}$).

To ensure a smooth transition in the joint space, we used the method from [30]. To assure the feasibility of the optimal solution and to lighten the burden of the deployment on the real robot we introduce a set of constraints. For this experiment we set $n_C = n_{IC} = 93$ inequality constraints: joint position limits, joint torque limits and a dynamic balance constraint to assure that after the t_{switch} the Zero Moment Point (ZMP) is located inside the support polygon of the robot.

A. Optimization for the simulated iCub in Gazebo

To obtain an optimal CoM trajectory that never violates the constraints we apply (1+1)-CMA-ES with CCA to our framework. As described in Section III-D, to avoid stalling, the algorithm requires a starting point that verifies the constraints, but for our application scenario it is not straightforward to set the right parameters that verify all the constraints at once.

Therefore we split the optimisation problem in two. In the first, we run an unconstrained optimisation problem, where we want to minimize the constraints violation to 0 (no constraint violation). In the second, we run a constrained optimization problem to find a solution that satisfies our "true" cost function. The two problems have similar settings ($t_{switch} = 1.5$s, 3 RBFs networks with 5 RBFs each, for a total number of parameters equal to 15, $T = 4.5$s with a control step of 10ms) but different fitness functions.

The fitness function for the first problem, or bootstrap problem, (*unconstrained optimization*) is:

$$\phi_{uc} = \begin{cases} \hat{l}, & \text{if fallen} \\ \phi_{bs}(T, n_c) & \text{otherwise} \end{cases}$$

where ϕ_{bs} is the fitness function from Section III-D, and $\hat{l}$ is a penalty that we apply when the robot falls.

The fitness function for the second problem, (*constrained optimization*) is:

$$\phi_{co} = \begin{cases} \hat{l}, & \text{if fallen} \\ -\frac{1}{\sum_i^3 w_i}\left[\frac{w_1 \sum_i^T |\mathbf{p}_{com}(i) - \mathbf{p}^*_{com}|}{\varepsilon_{max}} + \frac{w_2 \sum_i^T \mathbf{u}^2(i)}{u_{max}} + \frac{w_3 \sum_i^T b(i)}{b_{max}}\right], & \text{otherwise} \end{cases}$$

where $\phi_{co} \in [-1, 0]$, w_i are fixed weights, $\varepsilon_{max}, u_{max}, b_{max}$ are normalization factors for, respectively, the CoM position error (where $\mathbf{p}_{com}(i)$ is the current CoM position and $\mathbf{p}^*_{com}$ is the desired CoM), the effort penalty ($\mathbf{u}(i)$ is the torque at time i) and the backward penalty. The latter is defined as

$$b(i) = \mathbb{1}_{\{\mathbf{p}^x_{com}(i-1) > \mathbf{p}^x_{com}(i)\}}(\mathbf{p}^x_{com}(i-1) - \mathbf{p}^x_{com}(i)) + \quad (29)$$

$$\mathbb{1}_{\{\mathbf{p}^z_{com}(i-1) > \mathbf{p}^z_{com}(i)\}}(\mathbf{p}^z_{com}(i-1) - \mathbf{p}^z_{com}(i)) \quad (30)$$

where p^x_{com}, p^z_{com} are the x and z coordinate of the CoM position. This term penalizes all the trajectories that produce backward movements along the x or z direction. We use it to minimize undesirable oscillations of the robot along the sagittal plane.

To find a solution for the first problem (bootstrap) we performed 500 rollouts of (1+1)-CMA-ES (without CCA) with an exploration rate equal to 0.1. The solution is used as the starting point for the second problem, which is solved with (1+1)-CMA-ES with CCA, with the same exploration rate. A typical fitness profile for ϕ_{co} after 500 rollouts is shown in Figure 4. For this experiment, the weights for the penalty terms are respectively $w_1 = 1.3$ for the CoM error, $w_2 = 1$ for the effort and $w_3 = 3$ for the backward penalty.

We compare the optimized solution with one that was hand-tuned by an expert. The hand-tuned solution lasts for 11s, with $t_{switch} = 4.53$s. Even if the hand-tuned solution has a smoother profile for the CoM, it makes the robot move

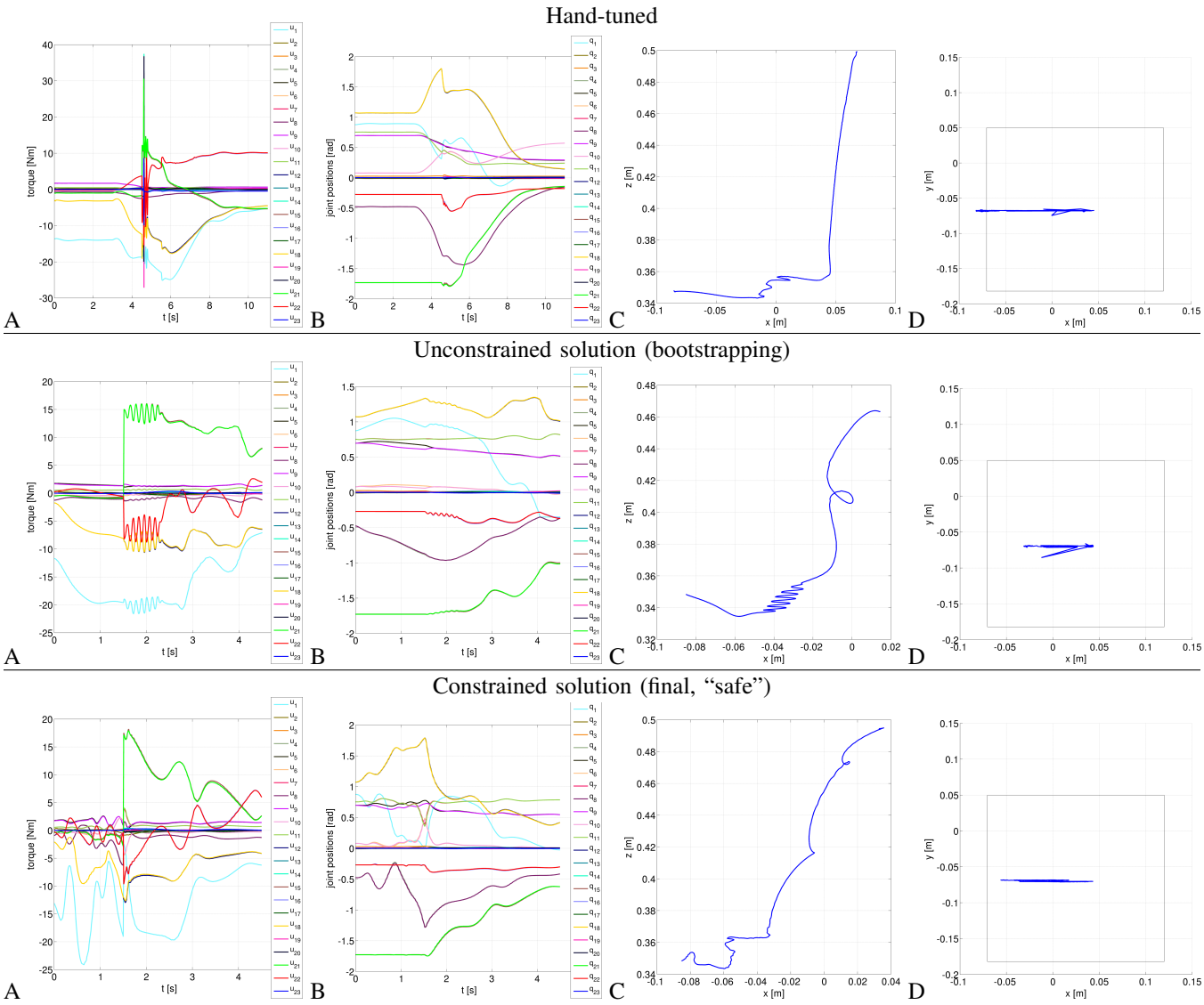

Fig. 3: Comparison of three typical different solutions. Rows: in the first row we show the results for a trajectory that is manually hand-tuned by an expert, while in second and third rows we show the final solutions after the bootstrapping and the final trajectory obtained by the constrained optimization. Columns: A plots the joint torques, B plots the joints positions during the experiment, C shows the CoM trajectory of the robot on the sagittal plane, D shows the evolution of the ZMP on the ground plane. In D, the square is a conservative estimation of the support polygon of the robot: for this reason, even if the ZMP of the hand-tuned experiment moves outside the support polygon, the robot does not fall. However, our optimization algorithm finds solutions that satisfy the conservative ZMP constraint.

faster during the stand-up and with higher torques (with a peak of torques at t_{switch}). The optimized trajectory, on the contrary, is faster (less than 5s), has lower energy consumption and lower CoM tracking error, as shown in Table II. By design, our solution satisfies all the problem constraints: as shown in Figure 3, it satisfies a very conservative constraint on the ZMP, that the hand-tuned solution was violating even if it was generating a physically feasible movement.

V. CONCLUSIONS

In this paper we propose a method to compute safe task trajectories for whole-body control of humanoid robots, where "safe" means that we ensure that the optimized trajectories lead to behaviors that never violate any of the problem/robot constraints. We propose to use (1+1)-CMA-ES with CCA, that we previously benchmarked in [5], to optimize the parameters of the task trajectories, while for controlling the robot we rely on a whole-body multi-task centroidal momentum controller from [6], [7].

We demonstrate our method on the iCub humanoid, to find a safe trajectory for the motion "stand-up from the chair", with multiple switching contacts and several constraints. By comparing the optimal solution with an hand-tuned one we showed that our method improves the task trajectories satis-

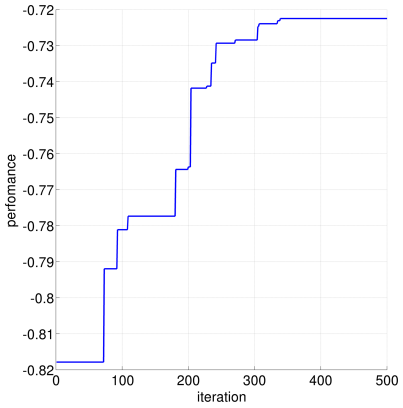

Fig. 4: Evolution of a typical fitness value for ϕ_{co}.

Solution	CoM error	effort penalty	backward penalty
hand-tuned	1.0	1.0	0.186292
unconstrained	0.806484	0.945829	0.962220
constrained	0.580518	0.646327	0.844451

TABLE II: Comparison of three typical solutions: hand-tuned, unconstrained (used for bootstrapping) and the final found by the constrained optimization. For each solution we separately compute each cost that composes the fitness ϕ_{co} in Equation IV-A. The constrained solution computed at the end of the whole optimization process has a better performance than the hand-tuned both in terms of COM error and total effort.

fying all constraints and reducing the energetic consumption. Realizing this kind of motions was not possible in our previous framework where task priorities were optimized [5].

Currently, the solution is optimized in simulation, as the number of roll-outs is too high for executing the algorithm directly on the real robot. However, the feedback controller and the good dynamics model make it applicable with minor adaptations to the real robot. In the future, we plan to leverage transferability approaches for deploying and refining the solution directly on the real robot[2] We plan to take inspiration from the "Intelligent Trial and Error" approach, which essentially suggests to generate a large diversity of high-performing trajectories offline and select the best one to test on the robot online by trial and error [31].

REFERENCES

[1] A. D. Prete, F. Nori, G. Metta, and L. Natale, "Prioritized motion force control of constrained fully-actuated robots: atask space inverse dynamics," *Rob. and Auton. Systems*, vol. 63, pp. 150 – 157, 2015.

[2] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *ICRA*, 2011, pp. 1283–1290.

[3] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, vol. 40, no. 1, pp. 17–31, Jan 2016.

[4] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi, "Learning soft task priorities for control of redundant robots," in *ICRA*, 2016.

[2]This means that our algorithm will be kept offline for the initialization of the starting solution, and will not run online on the robot: for that part, we will turn to data-efficient / micro-data learning algorithms.

[5] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi, "Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization," in *HUMANOIDS*, 2016.

[6] G. Nava, F. Romano, F. Nori, and D. Pucci, "Stability Analysis and Design of Momentum-based Controllers for Humanoid Robots. ," in *IROS*, 2016.

[7] D. Pucci, F. Romano, S. Traversaro, and F. Nori, "Highly dynamic balancing via force control," in *HUMANOIDS*, 2016, pp. 141–141.

[8] S. Ivaldi, J. Babič, M. Mistry, and R. Murphy, "Special issue on whole-body control of contacts and dynamics for humanoid robots," *Autonomous Robots*, vol. 40, no. 3, pp. 425–428, 2016.

[9] F. Flacco, A. De Luca, and O. Khatib, "Prioritized multi-task motion control of redundant robots under hard joint constraints," in *IROS*, Jan 2012, pp. 3970–3977.

[10] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The Int. Journ. of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[11] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *IROS*, 2011, pp. 4414–4419.

[12] J. Vaillant, A. Kheddar, and et al., "Multi-contact vertical ladder climbing with an hrp-2 humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 561–580, 2016.

[13] R. Lober, V. Padois, and O. Sigaud, "Efficient reinforcement learning for humanoid whole-body control," in *HUMANOIDS*, 2016.

[14] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 43:1–43:8, 2012.

[15] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments." in *ICAPS*, L. McCluskey et al., Ed. AAAI, 2012.

[16] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *ACC*, 2005, pp. 300–306.

[17] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IROS*, 2012, pp. 4906–4913.

[18] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *ICRA*, 2014.

[19] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Int. Conf. on Machine Learning*, 2009, pp. 1049–1056.

[20] R. Akrour, A. Abdolmaleki, H. Abdulsamad, and G. Neumann, "Model-free trajectory optimization for reinforcement learning," *CoRR*, vol. abs/1606.09197, 2016.

[21] P. Reist, P. Preiswerk, and R. Tedrake, "Feedback-motion-planning with simulation-based lqr-trees," *The Int. Journ. of Robotics Research*, vol. 35, no. 11, pp. 1393–1416, 2016.

[22] S. Traversaro, D. Pucci, and F. Nori, "A unified view of the equations of motion used for control design of humanoid robots," *On line*, 2017. [Online]. Available: https://traversaro.github.io/preprints/changebase.pdf

[23] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground." *Auton. Robots*, vol. 33, no. 4, pp. 399–414, 2012.

[24] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies." *Evolutionary Computation*, vol. 9, pp. 159–195, Jan 2001.

[25] D. V. Arnold and N. Hansen, "A (1+1)-CMA-ES for constrained optimisation," in *GECCO*, 2012, pp. 297–304.

[26] C. Igel, T. Suttorp, and N. Hansen, "A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies," in *GECCO*, 2006.

[27] L. Natale and et al., *The iCub Platform: A Tool for Studying Intrinsically Motivated Learning*. Springer, 2013, pp. 433–458.

[28] M. Fumagalli, S. Ivaldi, M. Randazzo, L. Natale, G. Metta, G. Sandini, and F. Nori, "Force feedback exploiting tactile and proximal force/torque sensing," *Aut. Robots*, vol. 33:4, pp. 381–398, 2012.

[29] F. Romano and et al., "A whole-body software abstraction layer for control design of free-floating mechanical systems," in *IEEE Int. Conf. on Robotic Computing (IRC)*, 2017, pp. 148–155.

[30] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," in *IROS*, 2010, pp. 1668–1674.

[31] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.